REMARKS

Claims 1-23 are currently pending in the present application. No claims have been amended in this response. Reconsideration of the claims is respectfully requested.

Applicant notes that formal drawings for this application were filed on 02/09/2001; however, the Office action did not acknowledge the receipt of those drawings nor whether the drawings were acceptable. Applicant kindly requests an acknowledgment and an indication of acceptability of the previously submitted formal drawings.

I. Summary of Present Invention

The present invention is a method for managing a given application type and, in particular, all instances of that application type. A manager object or entity is created to represent each application type running on a client. The manager object may reside anywhere in the distributed network, and the manager object is the target of management operations, which are then redirected to the appropriate client node. The distributed data processing system preferably includes a dataless management framework, also called a lightweight client framework (LCF). One advantage of the present invention is that a server management operation may be directed to a particular type or a given application instance without exposing the remainder of the client nodes to the operation.

The manager object preferably comprises a control routine and a registry containing a set of one or more elements in which each element is a dataset of context-specific information for a given application instance. The context information may include client node identity, installation location, e.g., a directory, installation identifier, e.g.,

database server name, or other administrative details. When a management operation, such as a query, is performed by a management server, the manager object redirects the query to the client node after possibly augmenting it with appropriate context information. Upon receipt of the query at the client, a query agent is started, and the context information is then used by a local query agent to identify which of the many installed instances of the application to target for the management operation.

II. 35 U.S.C. § 102(b)—Anticipation—*Tivoli Management Environment (Lightweight Client Framework or LCF)*

The Office action has rejected claims 1-23 under 35 U.S.C. § 102 as being anticipated by *TME 10 Framework Version 3.2: An Introduction to the Lightweight Client Framework* (abbreviated in the Office action as "ILCF"). Although a specific section of the statute is not recited, it is assumed that the rejection is based on 35 U.S.C. § 102(b) as the filing date of the present application is 12/18/1998 and the publication date of ILCF is October 1997. This rejection is respectfully traversed.

ILCF is a document that describes the Lightweight Client Framework (LCF) within the TME 10 Framework architecture; "TME" is an acronym for "Tivoli Management Framework". The Tivoli Management Framework includes a structured framework that provides core services and capabilities for management applications. One of the goals of the TME is to provide management capabilities for systems in a manner that is independent of the operating systems that are executing on those systems. The TME 10 framework is an object-oriented framework in which operations are carried out through interfaces defined for various objects. Operating system

functions can be encapsulated within various defined objects in order to represent system resources. Management of these resources is accomplished by accessing these objects and invoking requests, i.e. methods, on those objects. The TME 10 framework includes an implementation of a CORBA-compliant Object Request Broker (ORB), which is an object-oriented communication mechanism that allows methods to be invoked on objects in a manner that is independent of the location of those objects.

Given the above summary, ILCF does not disclose the claimed features of the present patent application, and this should be apparent to one of ordinary skill in the art after a careful reading of the present patent application. Although ILCF teaches managed regions, managed nodes, and an LCF in a manner similar to that described with respect to Figures 1-4 in the present patent application, ILCF does not teach the features of the claimed invention in the present patent application. The specification of the present patent application specifically states on page 10 that the present invention is preferably implemented within the Tivoli LCF: "The client component 24 preferably is dataless in the sense that the system management data is not cached or stored there in a persistent manner. This is a known product called the Tivoli lightweight client framework ('LCF')." In addition, Figures 1-4 are labeled as "Prior Art". Hence, the Office action formulates an anticipation rejection on a document that describes a system environment which the present patent application states is a preferable environment and admits as prior art.

However, the specification of the present patent application then continues with respect to Figures 5 and 6 to describe a novel invention that is not disclosed in ILCF, and

the rejection completely fails to provide any basis as to how ILCF discloses the present invention. More importantly, the rejection must fail because the features of the present invention are simply not disclosed in ILCF. For example, independent claim 1 of the present application, which is one of several independent claims that are rejected as anticipated by ILCF, reads as follows:

> 1. A method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:
>
> associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element include information representing a context of an application instance; and
>
> managing all instances of the application through the manager object.

ILCF does not disclose the features of "associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element include information representing a context of an application instance" nor "managing all instances of the application through the manager object".

This anticipation rejection includes multiple citations to portions of ILCF, none of which disclose the claimed features of the independent claims. In fact, the rejection of several of the claims merely applies portions of ILCF against claim elements without careful consideration of the claim language and the subject matter. While the form of the rejection appears appropriate, the argument presented by the rejection is illogical because the basis of the rejection, i.e. the cited portions of the reference, do not disclose what the rejection purports that they disclose. More importantly,

the reference as a whole fails to disclose the claimed elements of the independent claims of the present invention.

Applicant does not so much as argue against the rejection as show that the reference is wholly inadequate for supporting the rejection by juxtaposing the cited portions of the reference against the claim elements to which they have been applied by the rejection. For example, against the first feature of independent claim 1, the rejection cites "[Fig 2 page 7, page 10]". Page 7 states the following:

> When methods are invoked to access the distributed database, the database objects will access the local files or memory representing the database and, if required, communicate to objects representing the databases on other nodes to perform the requested operation. Again, all of this is transparent.
>
> 1.1.4 Basic Management Functions
>
> The TME 10 Framework is just that, a framework. In general, it primarily exists to provide enabling mechanisms for management applications. That is, the framework performs all of the necessary object resolution, communication and database access required so that the application developer can concentrate on providing management function rather than being bogged down with the details of cross-system communication and services.
>
> If we think in basic terms about the requirement of a management framework, it must provide only a few core capabilities such as data transfer and remote command execution. In practice, of course, it must also provide security, transaction support and other services required of any robust application environment. All of these capabilities are built into the objects that make up the TME 10 Framework.

Page 10 states the following:

> The TMR Server is responsible for maintaining the database and object registry. Recall that the database is distributed and spread among all various managed nodes in a TMR, but the TMR 10 Server provides the overall coordination of the distributed database.
>
> The Managed Node is a client that also runs the TME 10 Framework, including the client portion of the distributed database managed by the TMR Server. This

database contains objects and name registries related to the client and is linked to the server database via the Object Request Broker (ORB). Managed Nodes run complete versions of the TME 10 Framework and support any applications written to the standard TME 10 interfaces. In fact, the primary difference between the TME 10 Server and a Managed Node is that the server has overall control of the database. Otherwise, the full TME 10 Framework is installed and is functional on a Managed Node.

The PC Managed Node is an object in a Managed Node that represents a PC that is to be managed. The PC that is to be managed does not run the TME 10 Framework as described thus far. Instead, it contains and runs an agent that communicates with the PC Managed Node object on a Managed Node. The communication between the agent and the PC Managed Node object uses either TCP/IP or Netware's IPX protocol. Management applications interact with the PC Managed Node object, which then communicates with the agent running on the managed PC.

Clearly, the cited portions of ILCF do not disclose the first element of claim 1, i.e. "associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element include information representing a context of an application instance".

Against the second feature of independent claim 1, the rejection cites "[page 11]". Page 11 states the following:

The PC Managed Node concept allows large numbers of PCs to be managed without requiring the full TME 10 Framework to be installed on each system. However, because the agent and the PC Managed Node do not communicate via the ORB mechanism, all of the functions that can be performed by a PC Managed node are included in the agent module. This set of functions is not extendable by the customer and provides support for only a small subset of TME 10 management applications.

1.2.1 Interconnecting TMRs

Due to performance considerations mostly related to the distributed database (a database on every Managed Node), a TMR is limited supporting a maximum of about 200 Managed Nodes in typical environments. There can be many more PC Managed Nodes, but due to the limited functions supported by PC Managed Nodes, this option is not always

acceptable. However, for most companies a limit of 200 system that can be fully managed is nowhere near adequate.

To allow for the management of many thousands of systems, the TME 10 Framework allows for the interconnection of TMRs. Each TMR still maintains its own database distributed between the TMR Server and the Managed Nodes that make up the TMR, but management applications can perform their functions across TMR boundaries transparent to the application. Once the TMRs are interconnected, the TMR Servers for each TMR exchange information about the managed objects in each TMR and the framework handles routing requests to the TMR Server in the appropriate TMR. The owning TMR Server then passes the request (method invocation) on to the managed object.

The design of a managed environment with regard to the number of TMRs, the number of systems in each TMR and how the TMR boundaries are defined (organizationally, geographically, and so on) is something that each customer must decide for themselves. Though there may be good reasons to create multiple TMRs in a single environment (such as for security considerations), there are many cases where multiple TMRs have been used simply due to the limited number of Managed Nodes that can be supported by a single TMR Server.

Again, the cited portion of ILCF clearly does not disclose the claim element against which it is applied, namely the second element of claim 1, i.e. "managing all instances of the application through the manager object".

The arguments that have been given above for independent claim 1 are applicable to the claims that are dependent upon claim 1 as incorporating the elements of claim 1. However, Applicant provides the following additional arguments against some of the dependent claims. With respect to dependent claim 3, i.e. "wherein the dataless management framework includes a local agent that is controlled by the manager object to manage the application instance", the rejection recites "page 10" against this claim. However, as can be seen by the recitation of page 10 provided above, ILCF does not disclose the management of an application instance in the claimed manner.

With respect to dependent claim 5, i.e. "wherein the element includes information identifying a directory where the application instance is installed", the rejection recites "page 48" against this claim. Page 48 of ILCF describes various ways of installing the TME, e.g., "For OS/2, run install.exe from the \PC\LCF\OS2\CDROM directories." However, neither page 48 nor any other portion of ILCF discloses that the directory is identified by information in an element within a registry of a manager object for an application type on a given client, as is required by the consideration of dependent claim 5 along with independent claim 1 from which dependent claim 5 depends.

With respect to dependent claim 6, i.e. "wherein the element includes information identifying a name of a resource where the application instance is installed", the rejection recites "page 10" against this claim. However, as can be seen by the recitation of page 10 provided above, ILCF does not disclose the management of an installed application instance in the claimed manner.

With respect to dependent claim 7, i.e. "wherein the application type is discovered by the manager object", the rejection recites "page 118" against this claim. This is completely unintelligible as ILCF does not disclose a discovery process in any manner whatsoever, particularly in the claimed manner of claim 7.

With respect to dependent claim 8, i.e. "further including the step of discovering the application type prior to associating the manager object", the rejection recites "page 10" against this claim and states that the feature is an "inherent feature of registry". Applicant asserts that the rejection is entirely unclear as the claimed feature is not an

inherent feature of the registry in ILCF nor of registries in general, particularly in the claimed manner of claim 8.

The rejection has grouped together claims 10-23 as containing "similar limitations set forth of method claims 1-9". Applicant maintains that the arguments that have been given above also apply to similar claims in the group of claims 10-23.

Clearly, the rejection has not carefully considered the elements of the claims nor has the rejection pointed out the claimed features within ILCF as is required for a proper anticipation rejection. As stated at MPEP § 2131: "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Hence, the rejection of claims 1-23 is improper, and Applicant requests that the rejection be withdrawn.

III. 35 U.S.C. § 102(e)–Anticipation–*Lejeune et al.*

The Office action has rejected independent claims 1, 10, 17, 22, and 23 under 35 U.S.C. § 102(e) as being anticipated by Lejeune et al., "System for Managing and Processing Distributed Object Transactions and Process Implemented by said System", U.S. Patent No. 6,101,527, issued 08/08/2000 (hereinafter Lejuene et al.). This rejection is respectfully traversed.

Review of the teachings of the *Lejeune et al.* reference

The system disclosed in Lejeune et al. is fairly described in the "Summary of the Invention" section in column 3, line 50, to column 4, line 14:

> The object of the present invention is to eliminate the drawbacks of the prior art and to offer a system for managing and processing transactions that is efficient and simple to use.
>
> To this end, the system for managing and processing transactions mentioned in the preamble is noteworthy in that it achieves the implicit integration of resource managers adapted to the predefined interface, so as to integrate the participation of existing or future resource managers into a distributed transaction managed by the transaction manager by providing objects capable of participating in the transaction validation protocol used by the transaction manager, which objects address the resource managers through the predefined interface.
>
> Thus, when a server uses a resource through a resource manager (for example a data base), the participation of this data base in the transaction is managed directly by the present system, which means that this data base is encapsulated, and all the calls to the predefined interface are executed implicitly and are hidden from the programmer of the application, who consequently does not have to worry about them.
>
> Advantageously, in order for several servers to be able to use the same data base, each server comprises a specific local component which encapsulates the calls to the predefined interface in the form of objects called resource objects, while moreover one server for managing the predefined interface is provided per domain for implementing the encapsulation of the transaction validation protocol, thus allowing multiple distributed objects to execute multiple requests in the same single transaction.

Review of the rejection of claims 1, 10, 17, 22, and 23

As should be apparent from the above portion of Lejeune et al., the system is directed to a distributed transaction manager. Applicant maintains that it is difficult to understand how the applied reference is relevant to the

present invention. While Lejeune et al. discusses resource managers, management servers, registration operations, and ORBs, these similarities are irrelevant as Lejeune et al. fails to disclose the claim elements of the present invention.

Unfortunately, the use of Lejeune et al. against the present claims is as inadequate as the rejection under ILCF. The anticipation rejection under Lejeune et al. includes multiple citations to portions of Lejeune et al., none of which disclose the claimed features of the independent claims. In fact, the rejection of several of the claims merely applies portions of Lejeune et al. against claim elements without careful consideration of the claim language and the subject matter. In a manner similar to that described above with the rejection under ILCF, the form of the rejection appears appropriate, but the argument presented by the rejection is illogical because the basis of the rejection, i.e. the cited portions of the reference, do not disclose what the rejection purports that they disclose. More importantly, the reference as a whole fails to disclose the elements of the independent claims of the present invention.

The following is the rejection of independent claim 1 in its entirety:

> As per claim 1, Lejeune discloses a method of managing a set of clients in a distributed computer network having a management server [abstract, col 3 line 50-col 4 line 47], comprising the steps of:
>
> associating a manager object to each application type on a given client [col 5 line 20-col 6 line 36, col 7 lines 45-col 8 line 16], the manager object including a registry having a set of one or more elements [col 7 lines 25-37, col 8 lines 34-55], wherein each element includes information representing a context of an application instance [col 6 line 37-col 7 line 44]; and
>
> managing all instances of the application through the manager object [Fig 2].

Again, Applicant does not so much as argue against the rejection as show that the reference is wholly inadequate for supporting the rejection by juxtaposing the cited portions of the reference against the claim elements to which they have been applied by the rejection. For example, against a portion of the first feature of independent claim 1, i.e. "associating a manager object to each application type on a given client", the rejection cites "[col 5 line 20-col 6 line 36, col 7 lines 45-col 8 line 16]", which is more than two columns of patent text that supposedly disclose this feature. The portion at column 5, line 20, to column 6, line 36, states the following:

> X/OPEN also defines the XA interface between the transaction manager and the resource manager. This XA interface allows the involvement and the cooperation of heterogeneous resource managers in a single distributed transaction and adheres to a two-phase commit protocol managed by the transaction manager. A few instructions or routines used by this XA interface are explained below, it being understood that two types of routines are used. A first type allows a resource manager to call a transaction manager, a transaction manager in conformity with the X/OPEN model being assumed to provide the routines of this first type for allowing the resource managers to dynamically control their participation during the initiation of transactions, thus:
>
> ax_reg makes it possible to register a resource manager through a transaction manager,
>
> ax_unreg allows the transaction manager to clear the registration of a resource manager.
>
> The second type of routine relates to routines provided by the resource managers and called by the transaction managers in order to inform these resource managers of the initiations of transactions, thus:
>
> xa_close makes it possible to disconnect from the resource manager,
>
> xa_commit makes it possible to request a resource manager to validate a transaction,
>
> xa_end makes it possible to dissociate an execution unit from a transaction,
>
> xa_forget allows the resource manager to forget the knowledge it has of a transaction completed heuristically,

xa_open makes it possible to connect to the resource manager and initialize it for use by an application program,

xa_prepare makes it possible to request the resource manager to prepare to validate a transaction,

xa_recover makes it possible to obtain from the resource manager a list of transaction identifiers it has prepared or completed heuristically,

xa_rollback makes it possible to request a resource manager to cancel a transaction,

xa_start makes it possible to start or restart a transaction; it also associates an identifier with a future task that the execution unit requests from the resource manager, thus making it possible to indicate to the resource manager that the future tasks will be executed on behalf of a transaction.

In order to access this second type of routine, the transaction manager uses a structure called xa_switch_t, whose name is published by the resource manager. This structure contains all the entry points to the routines of the second type as well as information relative to the resource manager such as its name. However, a resource manager knows that certain required tasks are executed on behalf of a given transaction from the fact that the calls from the transaction manager and from the application program are sent to the same execution control unit, and for the utilization of certain routines of the second type, this imposes the following restraints:

at the opening and the closing of the resource managers, the calls to the routines xa_open and xa_close must be executed for each execution control unit accessing the resource manager, and the call to the routine xa_open must be executed before sending any other call to a routine of the second type. All associations of an execution control unit with transactions must be completed by the transaction manager before it dissociates the resource manager from the application by calling the routine xa_close;

when execution units are associated with transactions, the calls to the routines xa_start and xa_end must be executed from the same execution control unit that accesses the resource manager.

as for the calls to routines of the second type participating in the two-phase commit protocol, only one call to the routine xa_prepare is allowed for requesting the resource manager to prepare to validate the work executed on behalf of a transaction, the subsequent calls of this routine resulting in an error, and all associations of execution control units with the transaction, that is the processes involved in the transaction, must be completed before the system can request the resource manager to prepare the validation.

The portion at column 7, line 45, to column 8, line 16, states the following:

In order to manage a transaction, the client applications CL can use two types of context management: direct context management, with which the client application manipulates the objects of the transaction service TS by directly invoking the objects which represent the transaction, and indirect context management, with which the client application does not see the objects of the transaction service TS, which are manipulated internally by the system according to the invention. Moreover, a transaction object can require that information related to transactions be propagated either implicitly or explicitly. In the implicit propagation mode, the transaction context is propagated with the request in a transparent way through the ORB, which mode involves cooperation between the ORB and the transaction service TS, whereas in the explicit propagation mode, the information related to transactions appears in the form of explicit parameters given by the client. During transactions, the clients can use one or both forms of context management and can communicate with objects which may require either transaction propagation mode, the interfaces of the system allowing this switching between these two modes. However, in the majority of applications two combinations predominate. The first corresponds to the use of direct context management and an explicit transaction propagation mode, in which case the coordination, completion and control objects are manipulated directly at the level of the programming of the application and at least some of them are propagated explicitly as arguments; this combination makes it possible to develop transaction applications with implementations similar to a traditional CORBA

("Common Object Request Broker Architecture," a specification of the OMG and X/OPEN) application. The second corresponds to the use of indirect context management and an implicit transaction propagation mode, in which case the client application uses the current local object to create and control the transaction and to associate the transaction context with execution units. When the client sends a request to transaction objects, the transaction context associated with the current execution unit is implicitly propagated with the request.

Clearly, the cited portions do not disclose anything equivalent to "associating a manager object to each application type on a given client".

Against another portion of the first feature of independent claim 1, i.e. "wherein each element includes information representing a context of an application instance", the rejection cites "[col 6 line 37-col 7 line 44]". The portion at column 6, line 37, to column 7, line 44, states the following:

> The system for managing and processing transactions according to the invention also, of course, provides the general functionalities such as initiating or completing the transactions, controlling the propagation and the span of the transactions, allowing multiple objects to be involved in a single transaction, allowing objects to associate changes in their internal state with a transaction, or coordinating the completion of transactions by means of a two-phase (preparation and validation) commit and the ability to back out. This system for managing and controlling transactions thus defines interfaces which allow multiple distributed objects to execute multiple requests in the same transaction. The following definitions, with the aid of FIG. 1, will provide a better understanding the contents of the invention. Thus, within the context of distributed transaction applications, where a distributed transaction can involve multiple distributed objects executing multiple requests, a transaction client is a client CL that invokes various operations within the same transaction, while a client that initiates the transaction is called the requester or initiator of the transaction. The span of a transaction is defined by a

transaction context TC which is shared by all of the participating objects, the transaction context being propagated with each transaction request, the transaction objects TO of a transaction server TRS participating in the propagation of the transaction. At the level of a recoverable server RS, the recoverable objects RCO are transaction objects wherein the data are assigned by the validation ("commitment") or the cancellation ("rolling back") of the transaction, a recoverable object causes a so-called resource object RSO to be registered by the transaction service TS in order to associate the changes performed on this data with this transaction. The transaction service controls the two-phase commit protocol by sending requests to the registered resources. The interaction between these various components is indicated below with the aid of FIG. 1. The path ending in an arrow labeled 1 indicates that a transaction client CL initiates and completes the transaction. The path ending in an arrow labeled 2 indicates that a transaction context TC is created when a transaction is initiated and is transmitted to the transaction objects TO and to the recoverable objects RCO with each transaction request. The path marked with an arrow labeled 3 indicates that a transaction object TO propagates the transaction without being involved in the completion of this transaction, however it can force the cancellation ("rollback") of this transaction. In the path marked with an arrow labeled 4, a recoverable object RCO causes a resource object RSO to be registered by the transaction service TS in order to allow the completion of the transaction; it can also force the cancellation ("rollback") of this transaction. Finally, the path marked with an arrow labeled 5 indicates that a resource object RSO participates in the completion of the transaction according to the two-phase commit protocol controlled by the transaction service TS. In this way, complete interoperability is obtained between this functional model according to the invention and the X/OPEN distributed transaction processing model. Because of this total compatibility between these two models, the recoverable objects can be implemented using resource managers adapted to the XA interface, while the system for processing and managing transactions according to the invention implements the encapsulation of this XA interface, the implicit registration of the resource objects as well as the coordination of the recovery and the completion of the transactions. The system according to the invention associates a certain number of objects

with each transaction, which objects are used to form the transaction context TC and are preferably the following:

the coordination object, whose interface allows the recoverable objects to register resources for the completion of a transaction,

the completion object, whose interface supports the operations for validating or canceling the transaction,

or the control object, which allows the clients to retrieve both the coordination and completion objects.

Clearly, when considering the claim as a whole as is required by proper examination practice, the cited portion does not disclose anything equivalent to "wherein each element includes information representing a context of an application instance", where the element is in a registry of a manager object. At most, the rejection has succeeded in finding a reference that contains a term that is somewhat similar to a term that also appears in the claim: "the context of distributed transaction applications" in Lejeune et al. versus "a context of an application instance" in the present application. As is well known, however, such term matching is not a sufficient basis for supporting an anticipation rejection.

Against the second feature of independent claim 1, i.e. "managing all instances of the application through the manager object", the rejection cites "[Fig 2]". Applicant maintains that it is not possible for the cited figure to disclose "managing all instances of the application through the manager object" when the text of Lejeune et al. fails to disclose the manager object in the first feature of the claim.

The rejection has grouped together independent claims 10, 17, 22, and 23 as containing "similar limitations set forth of method claim 1". Applicant maintains that the arguments that have been given above for independent claim 1 also apply to claims 10, 17, 22, and 23.

Clearly, the rejection has not carefully considered the elements of these claims nor has the rejection pointed out the claimed features within Lejeune et al. as is required for a proper anticipation rejection. As stated at MPEP § 2131: "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Hence, the rejection of claims 1, 10, 17, 22, and 23 is improper, and Applicant requests that the rejection be withdrawn.

IV. 35 U.S.C. § 103(a)–Obviousness–*Lejeune et al. in view of Bereiter*

The Office action has rejected claims 2-9, 11-16, and 18-21 under 35 U.S.C. § 103(a) as unpatentable over Lejeune et al. in view of Bereiter. This rejection is respectfully traversed.

With respect to dependent claim 2, the rejection properly states that Bereiter discloses a dataless management framework. However, claim 2 depends from claim 1, and as argued above, Lejeune et al. does not disclose the features of claim 1. Hence, a combination of the teaching of Bereiter and Lejeune et al. cannot support a rejection of dependent claim 2.

With respect to dependent claim 3, the rejection states that the feature of "wherein the dataless management framework includes a local agent that is controlled by the manager object to manage the application instance" is an inherent

feature of a dataless management framework. First, Applicant notes that neither Lejeune et al. nor Bereiter disclose the feature of using a manager object to manage an application instance as claimed. Second, Applicant asserts that the teachings of two references that do not separately disclose a claimed feature cannot be combined to "inherently" produce a claimed feature. The rejection must explain which features of the teachings of each reference are being combined as support for an obviousness-type argument.

With respect to claims 4-9, the rejection points to portions of Lejeune et al. as showing features in these dependent claims. However, these claims depend only from independent claim 1, and the rejection applied only Lejeune et al. against claim 1 without resorting to Bereiter. Hence, it is improper for these claims to be rejected under 35 U.S.C. § 103(a); if all of the features of these claims are shown in Lejeune et al., then these claims should be rejected under 35 U.S.C. § 102 like claim 1.

With respect to dependent claim 5, i.e. "wherein the element includes information identifying a directory where the application instance is installed", the rejection recites "Lejeune col 9 line 63-col 10 line 18" against this claim. However, this portion of Lejeune et al. does not even mention directories.

With respect to dependent claim 6, i.e. "wherein the element includes information identifying a name of a resource where the application instance is installed", the rejection recites "Lejeune col 5 lines 55-65, col 6 lines 37-67" against this claim but also states that the feature is "inherent feature of an application context". Applicant asserts that the cited portion of Lejeune et al. does not disclose this feature, and it is confusing why an inherency argument is

provided if the feature is supposedly shown in the cited passage.

With respect to dependent claim 7, i.e. "wherein the application type is discovered by the manager object", the rejection recites "Lejeune col 5 lines 25-67" against this claim. Lejeune et al. does not disclose a discovery process in any manner whatsoever, particularly in the claimed manner of claim 7.

With respect to dependent claim 8, i.e. "further including the step of discovering the application type prior to associating the manager object", the rejection recites "Lejeune col 11 lines 49-67" against this claim and states that the feature is an "inherent feature of resource manager which manages the object". Lejeune et al. does not disclose a discovery process in any manner whatsoever, particularly in the claimed manner of claim 8. Again, it is confusing why an inherency argument is provided if the feature is supposedly shown in the cited passage.

The rejection has grouped together claims 11-16 and 18-21 as containing "similar limitations set forth of method claims 1-9". Applicant maintains that the arguments that have been given above also apply to similar claims in the group of claims 11-16 and 18-21.

Examiner bears the burden of establishing a *prima facie* case of obviousness.

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Only when a *prima facie* case of obviousness is established does the burden shift to the applicant to produce evidence of nonobviousness.

In re Oetiker, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993). If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). In response to an assertion of obviousness by the Patent Office, the applicant may attack the Patent Office's *prima facie* determination as improperly made out, present objective evidence tending to support a conclusion of nonobviousness, or both. *In re Fritch*, 972 F.2d 1260, 1265, 23 U.S.P.Q.2d 1780, 1783 (Fed. Cir. 1992).

With respect to claims 2-9, 11-16, and 18-21, Applicant respectfully submits that claimed features are not shown in the prior art references nor can the teachings of the references be combined to disclose the present invention. Hence, the rejection of claims 2-9, 11-16, and 18-21 does not establish a *prima facie* case of obviousness based on the prior art. Therefore, the rejection of claims 2-9, 11-16, and 18-21 under 35 U.S.C. § 103(a) has been shown to be insupportable, and these claims are patentable over the applied references. Applicant requests that the rejection be withdrawn.

In addition, as noted above, the rejection of most of the dependent claims under 35 U.S.C. § 103(a) is improper as these claims should have been rejected under 35 U.S.C. § 102.

V. Conclusion

It is respectfully urged that the present patent application is patentable, and Applicant kindly requests a Notice of Allowance.

For any other outstanding matters or issues, the examiner is urged to call or fax the below-listed telephone numbers to expedite the prosecution and examination of this application.

DATE: January 30, 2002

Respectfully submitted,

Joseph R. Burwell
Reg. No. 44,468
ATTORNEY FOR APPLICANT

Law Office of Joseph R. Burwell
P.O. Box 28022
Austin, Texas 78755
(512) 502-9448 (voice)
(512) 597-1218 (fax)